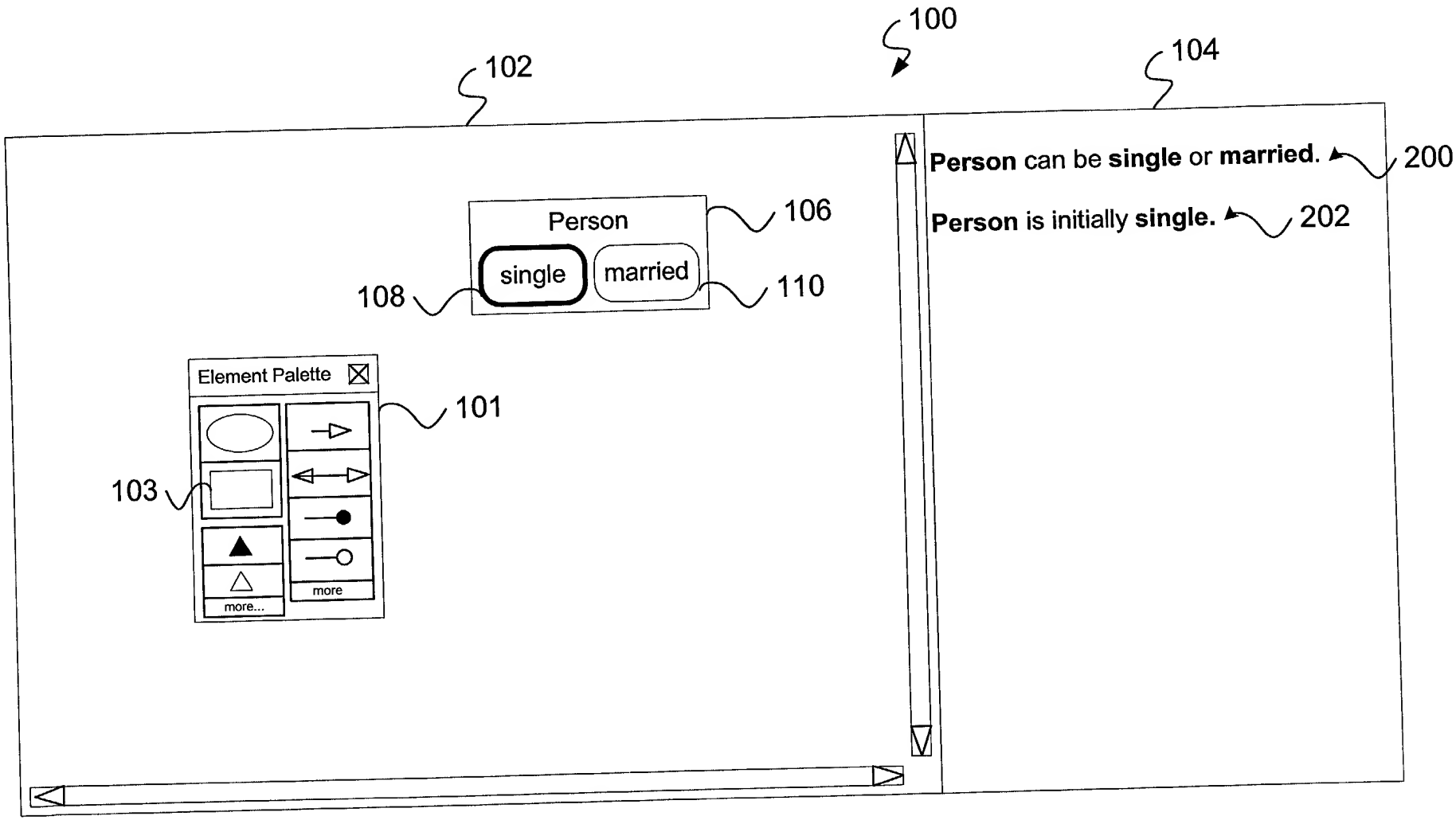

100
102
104
Person can be single or married.
200
Person is initially single.
202
Person
106
single
married
108
110
Element Palette
101
103
more
more...

FIG. 1

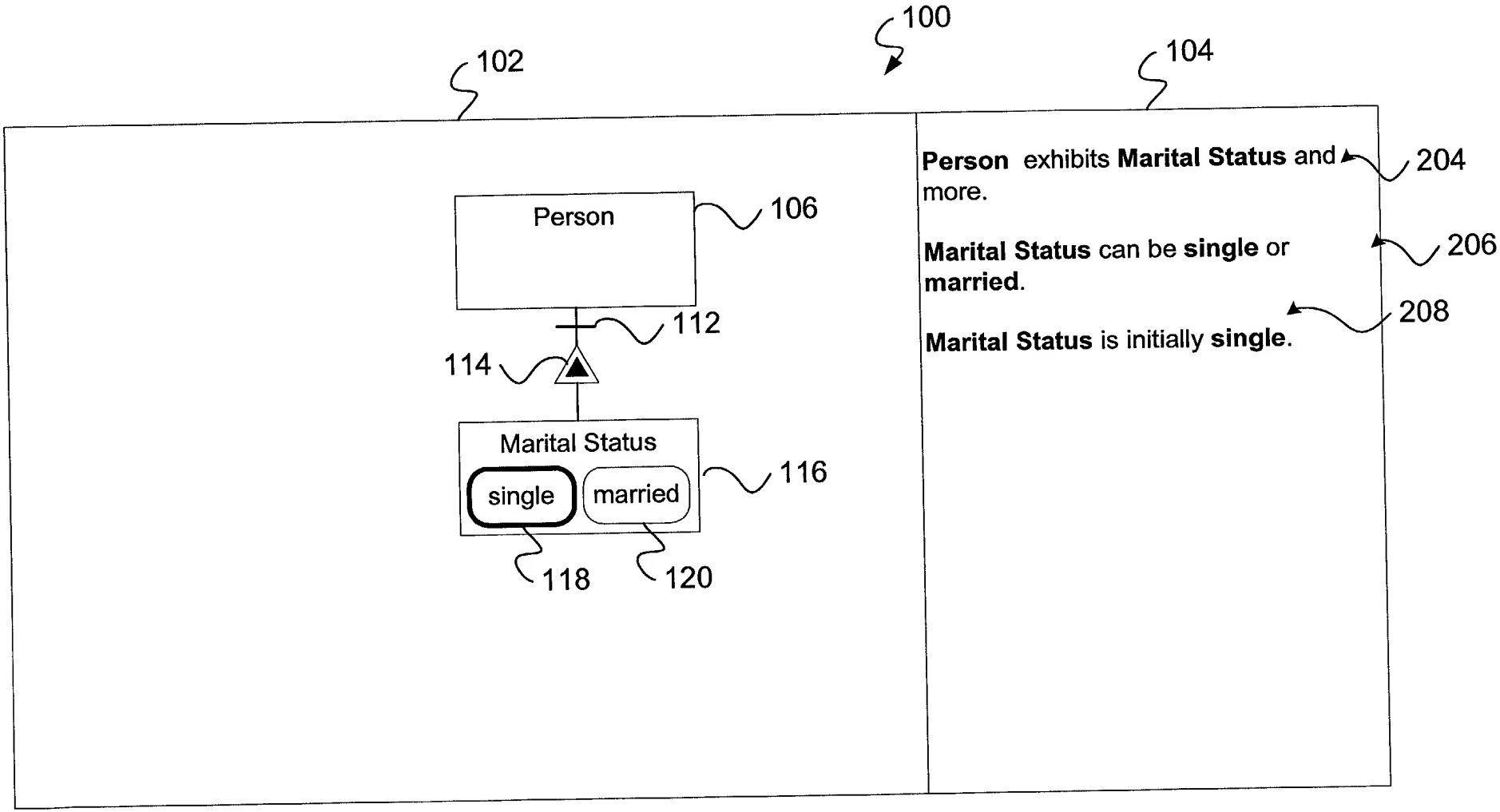

100
102
104
Person
106
112
114
Marital Status
single
married
116
118
120
Person exhibits Marital Status and more.
204
Marital Status can be single or married.
206
Marital Status is initially single.
208

FIG. 2

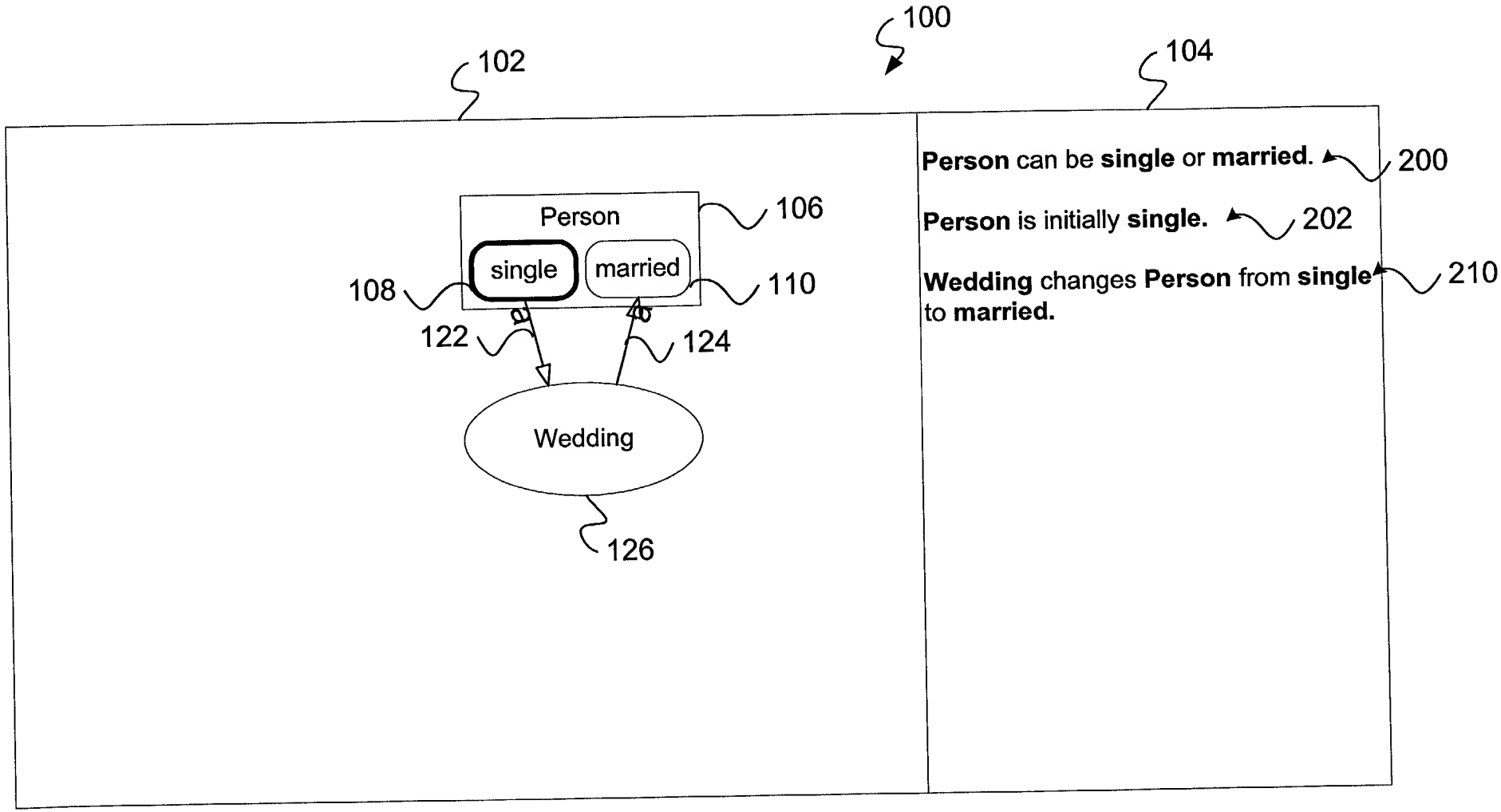

100
102
104
Person
106
single
married
108
110
122
124
Wedding
126
Person can be single or married.
200
Person is initially single.
202
Wedding changes Person from single to married.
210

FIG. 3

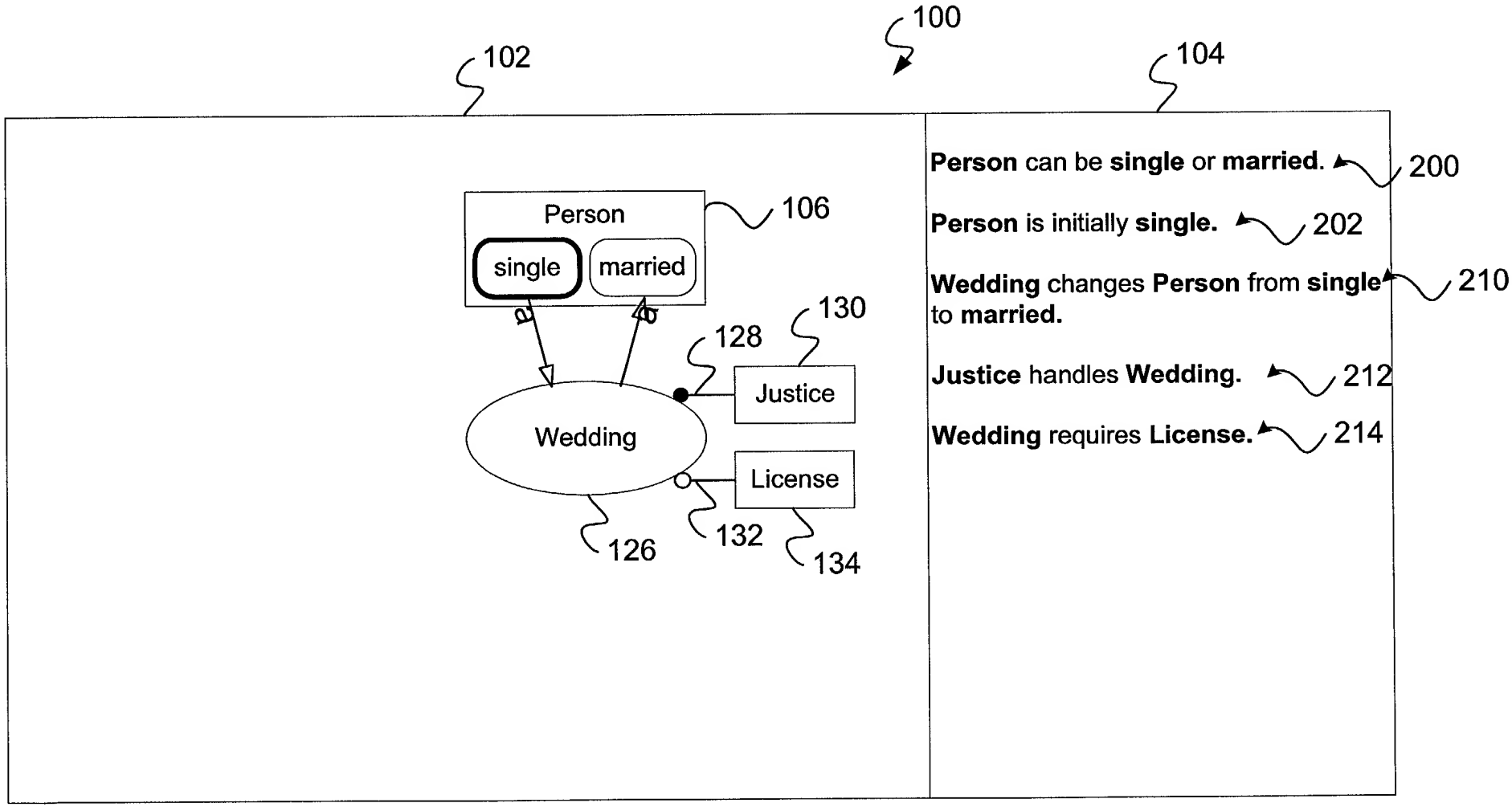

100
102
104
Person
106
single
married
130
128
Justice
Wedding
License
126
132
134
Person can be single or married.
200
Person is initially single.
202
Wedding changes Person from single to married.
210
Justice handles Wedding.
212
Wedding requires License.
214

FIG. 4

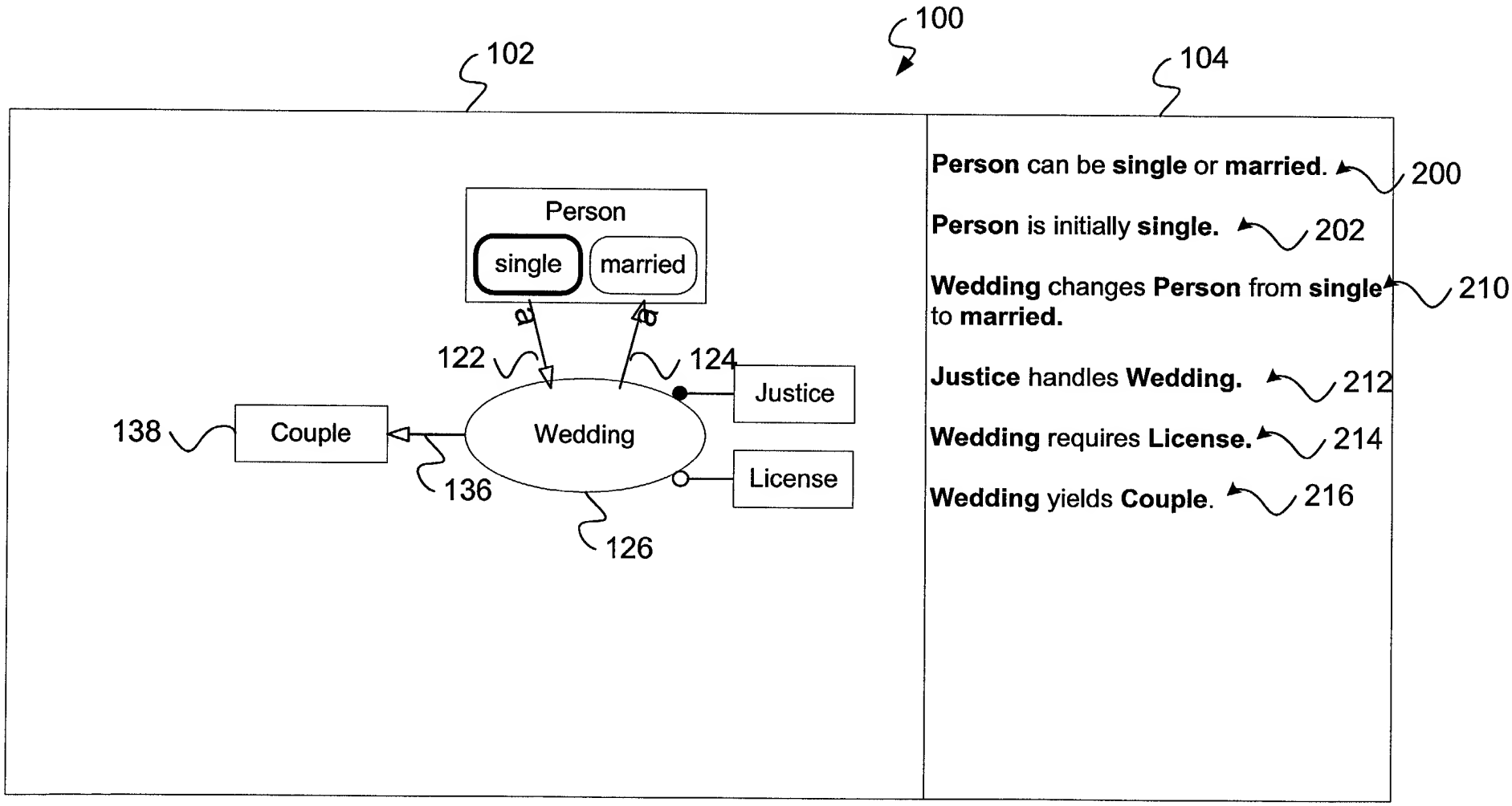

100
102
104
Person
single
married
122
124
Justice
138
Couple
Wedding
License
136
126
Person can be single or married.
200
Person is initially single.
202
Wedding changes Person from single to married.
210
Justice handles Wedding.
212
Wedding requires License.
214
Wedding yields Couple.
216

FIG. 5

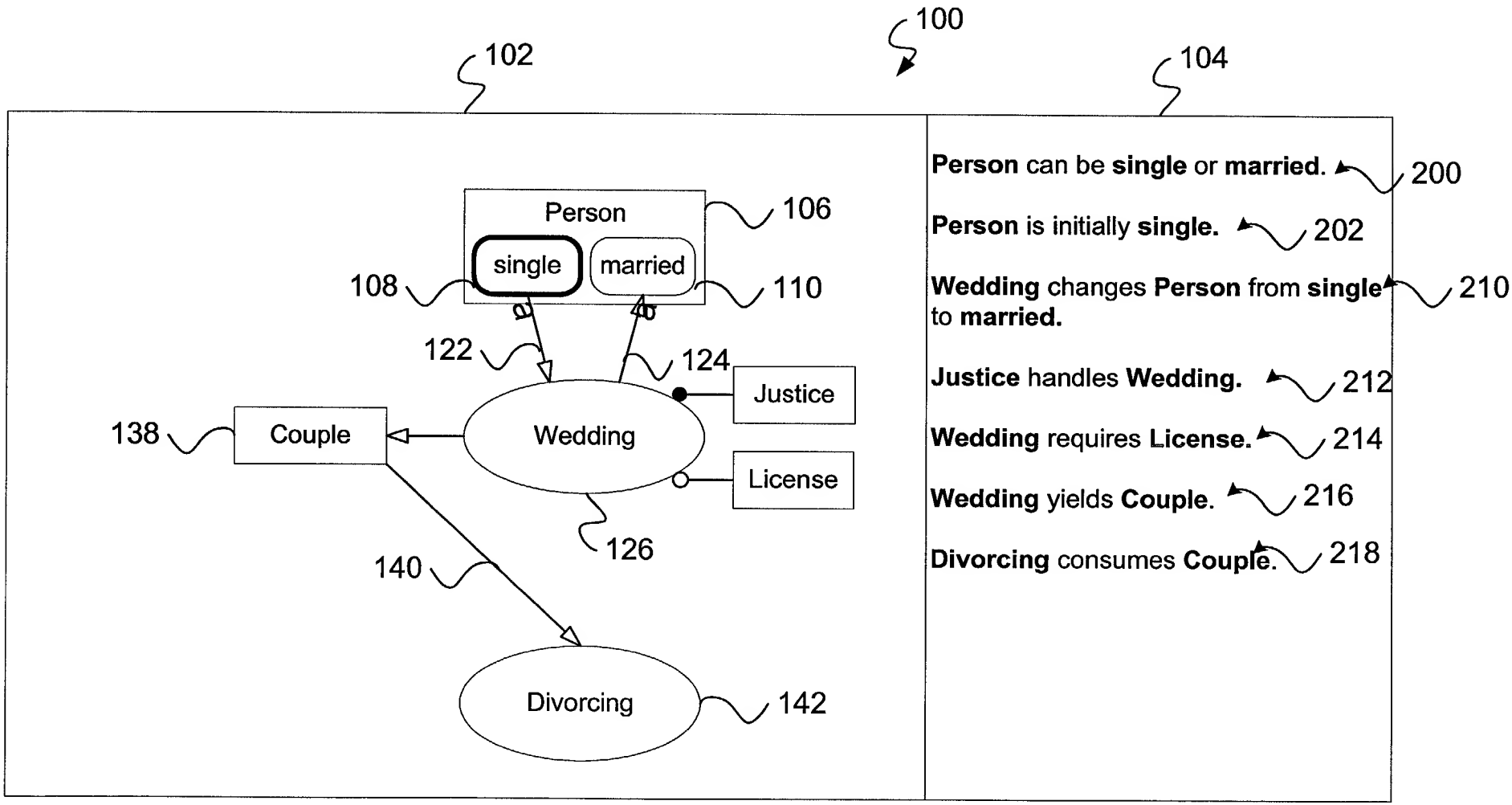

100
102
104
Person
106
single
married
108
110
122
124
Justice
138
Couple
Wedding
License
126
140
Divorcing
142
Person can be single or married.
200
Person is initially single.
202
Wedding changes Person from single to married.
210
Justice handles Wedding.
212
Wedding requires License.
214
Wedding yields Couple.
216
Divorcing consumes Couple.
218

FIG. 6

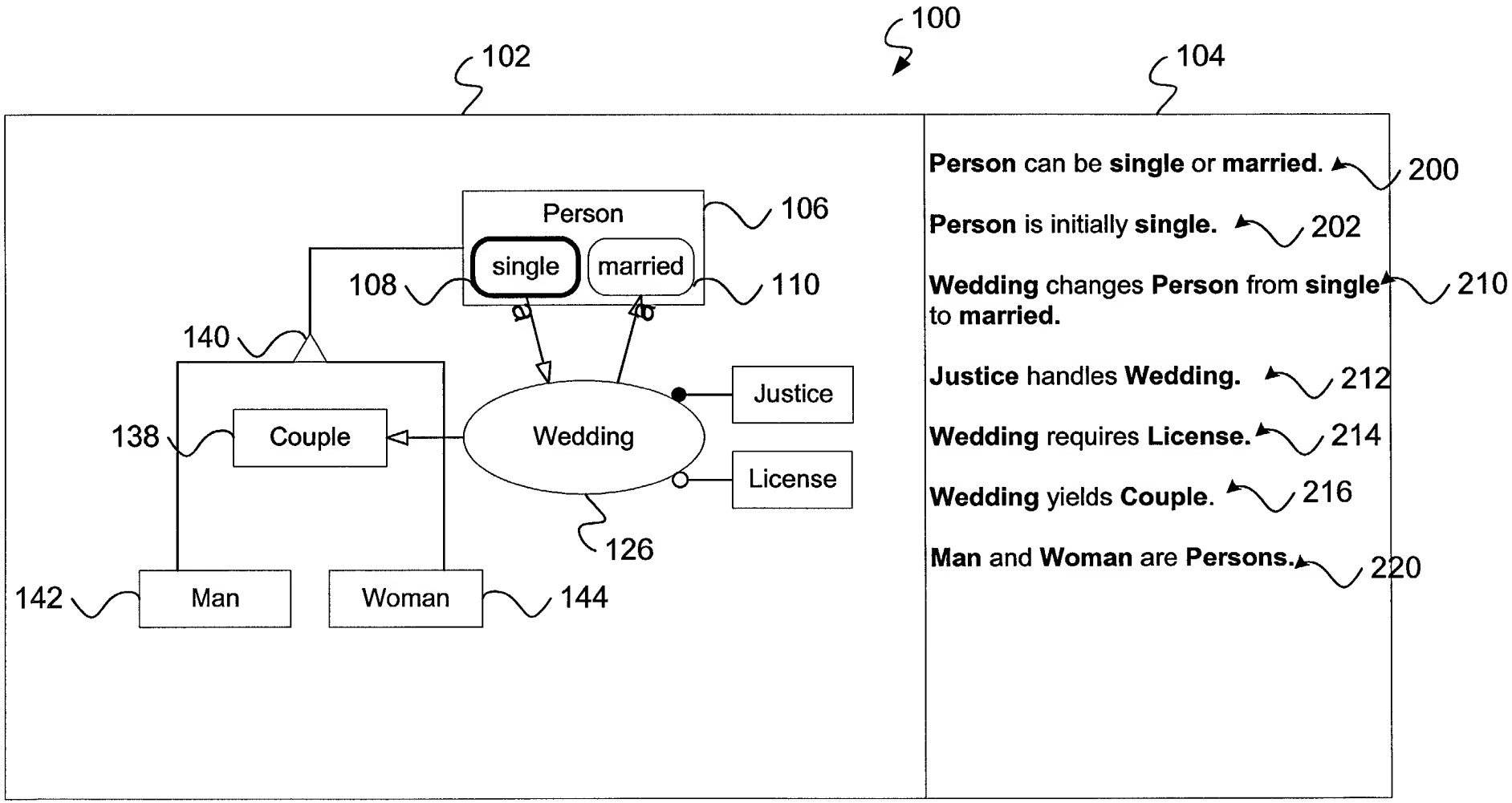

100
102
104
Person
106
single
married
108
110
140
Justice
Couple
Wedding
138
License
126
142
Man
Woman
144
Person can be single or married.
200
Person is initially single.
202
Wedding changes Person from single to married.
210
Justice handles Wedding.
212
Wedding requires License.
214
Wedding yields Couple.
216
Man and Woman are Persons.
220

FIG. 7

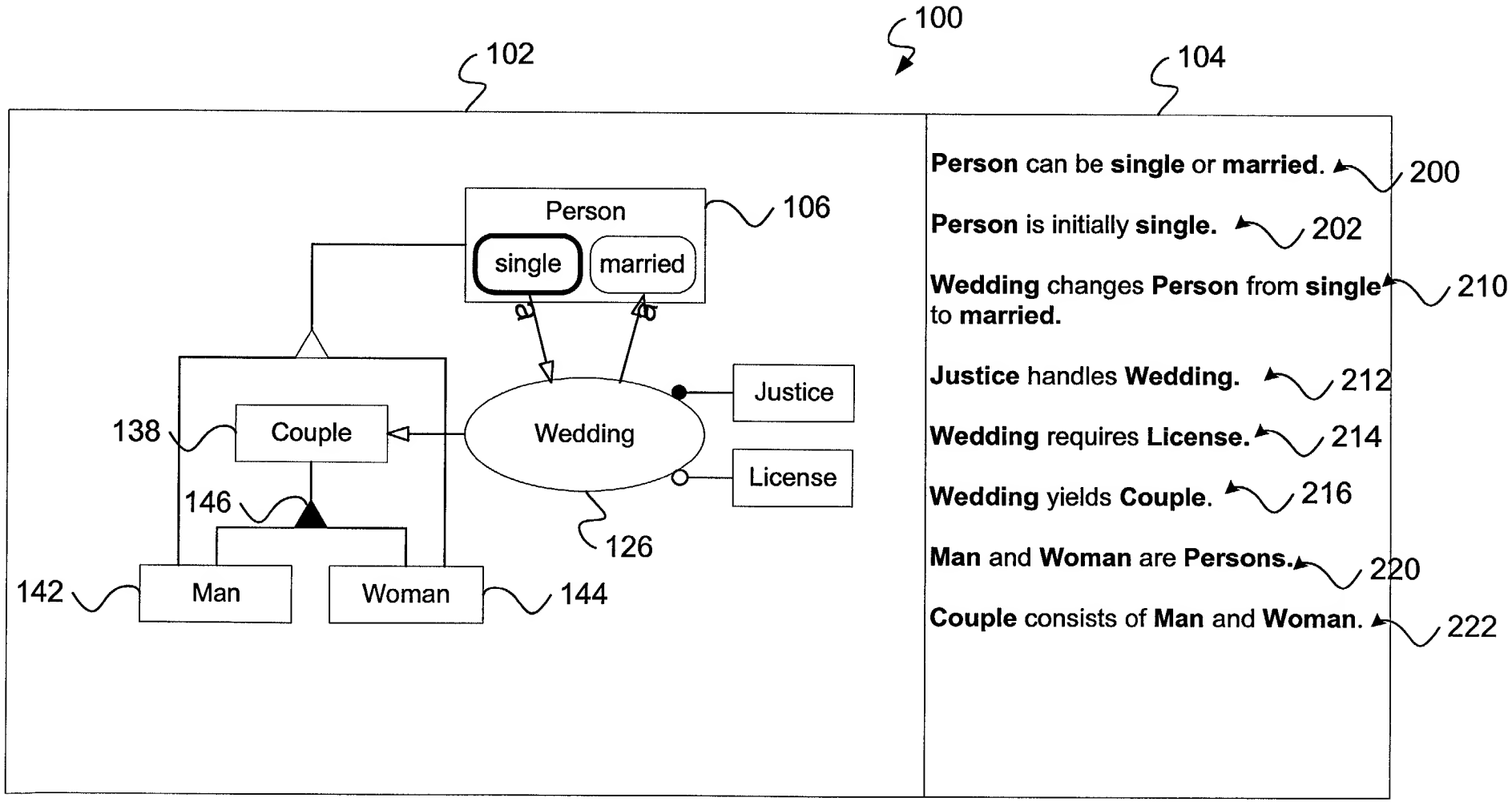


FIG. 8

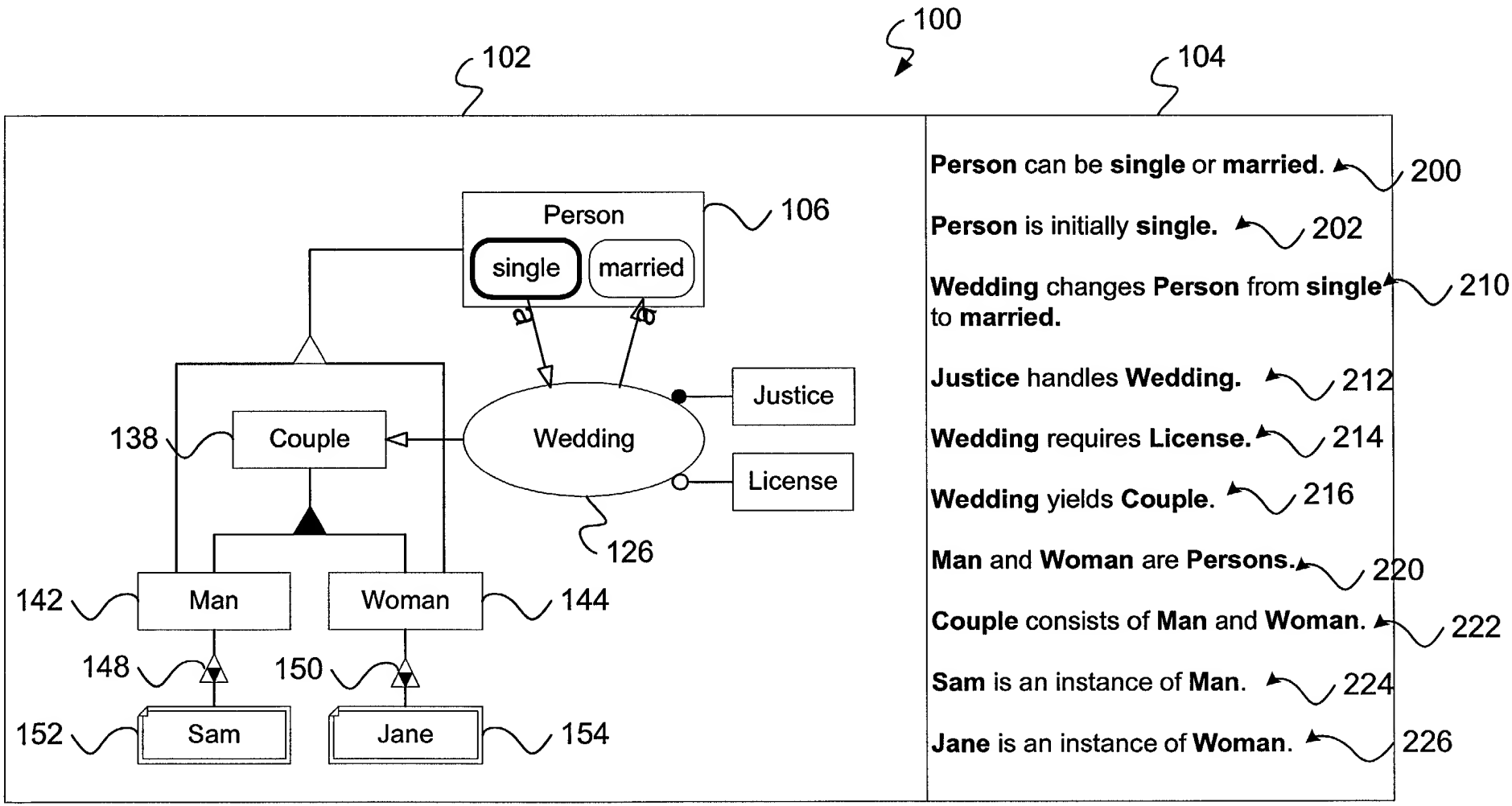


FIG. 9

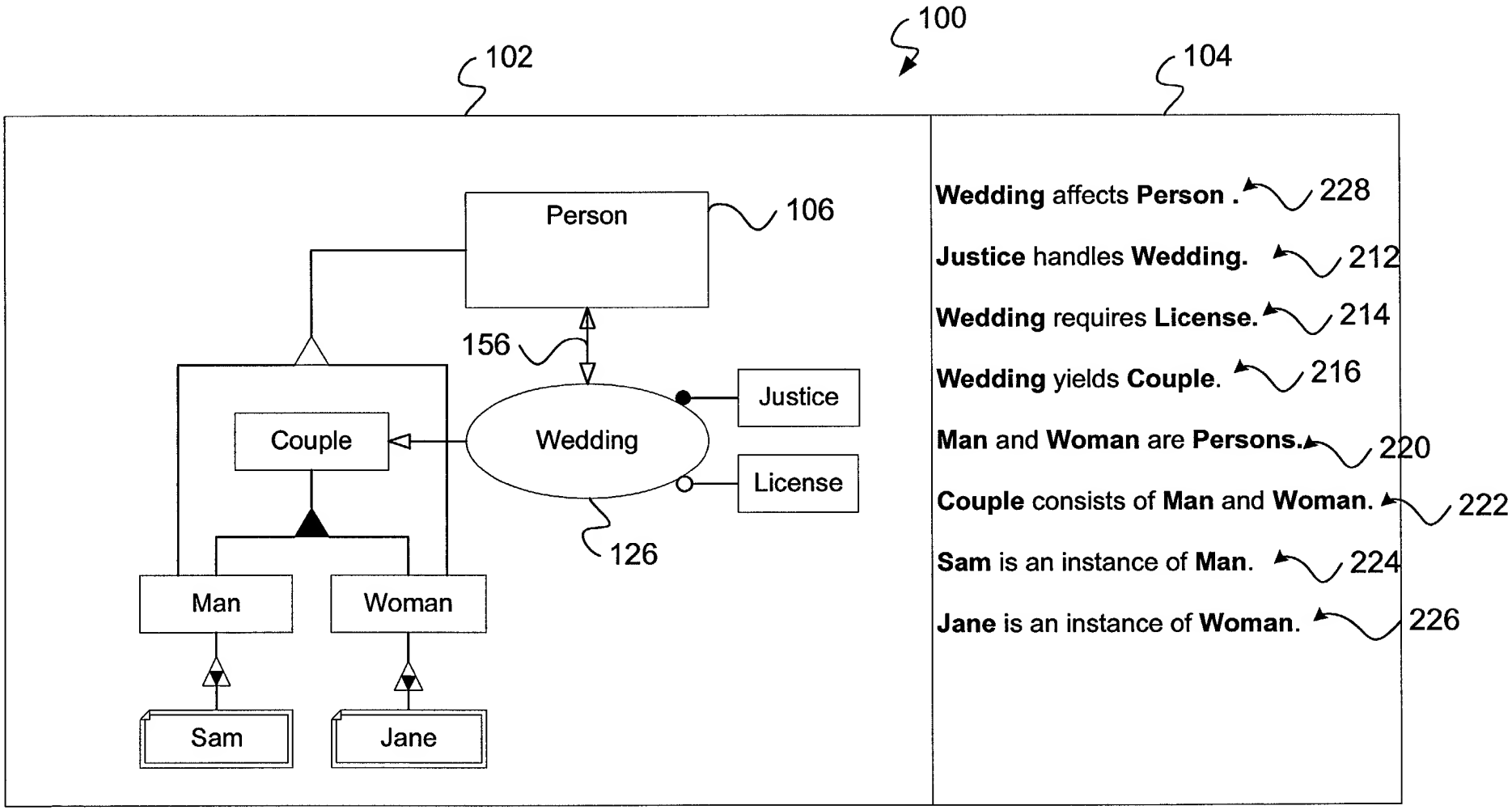


FIG. 10

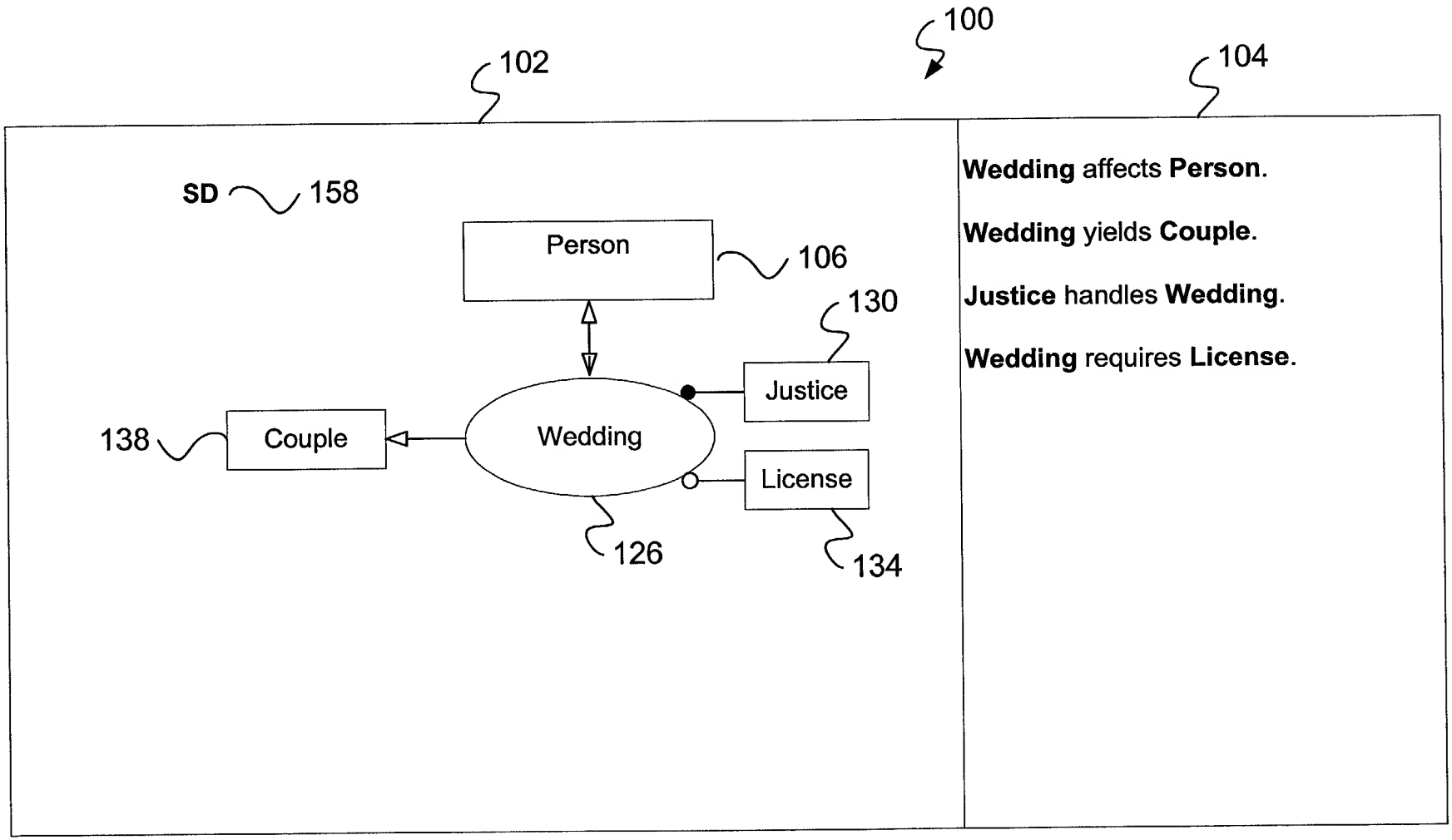

100
102
104
SD
158
Person
106
130
Justice
Wedding
138
Couple
License
126
134
Wedding affects **Person**.
Wedding yields **Couple**.
Justice handles **Wedding**.
Wedding requires **License**.

FIG. 11

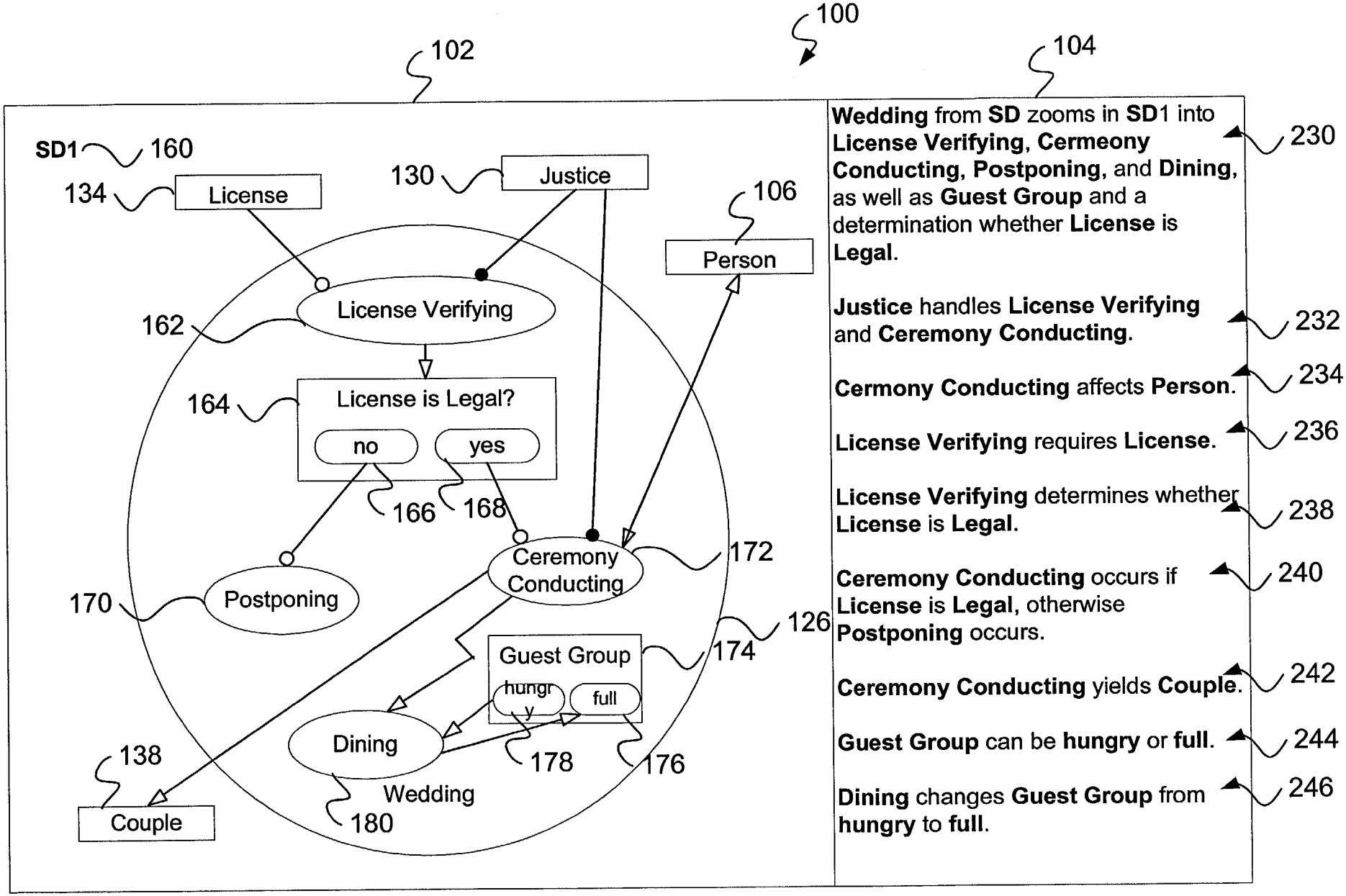


FIG. 12

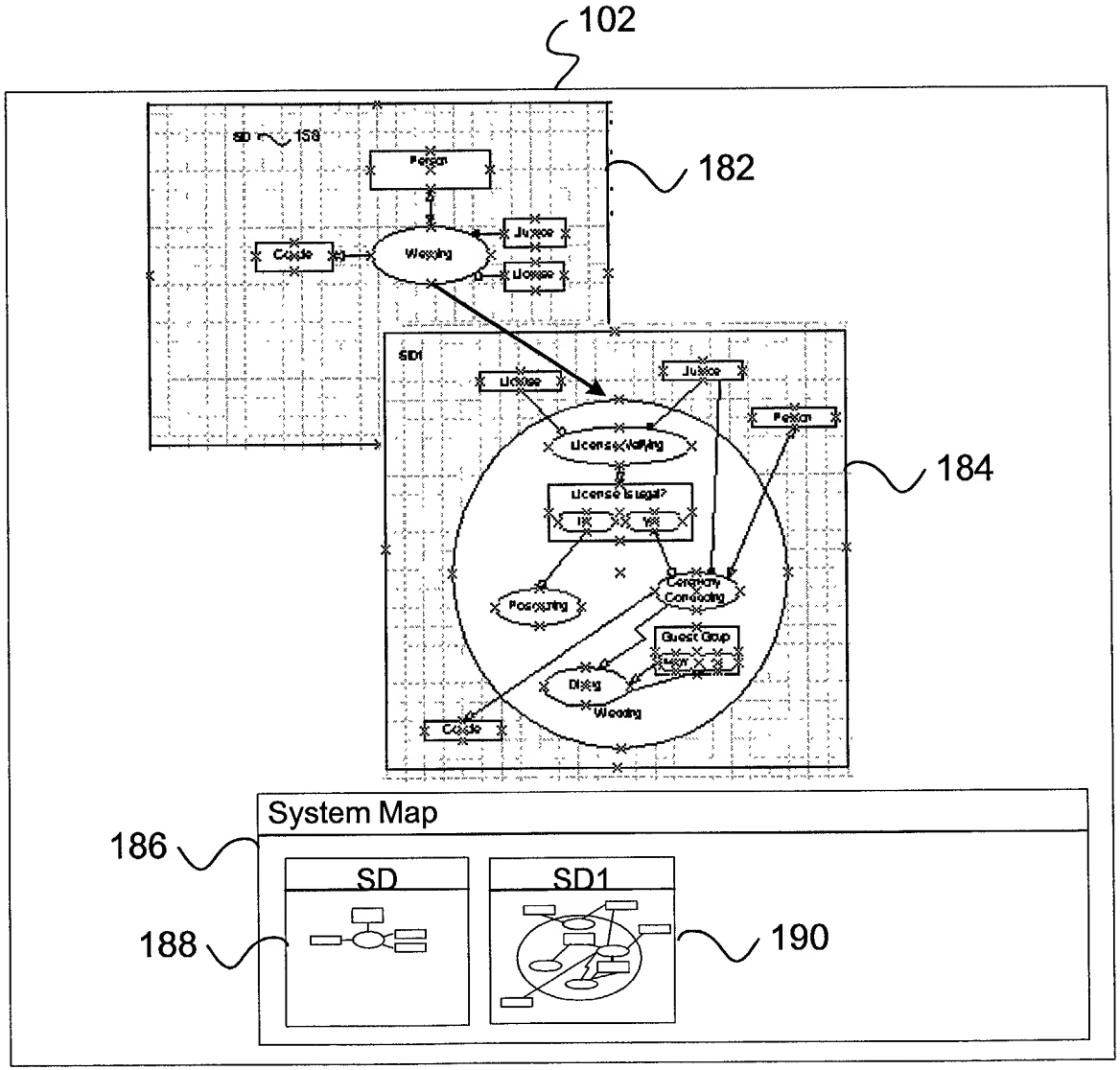


FIG. 13

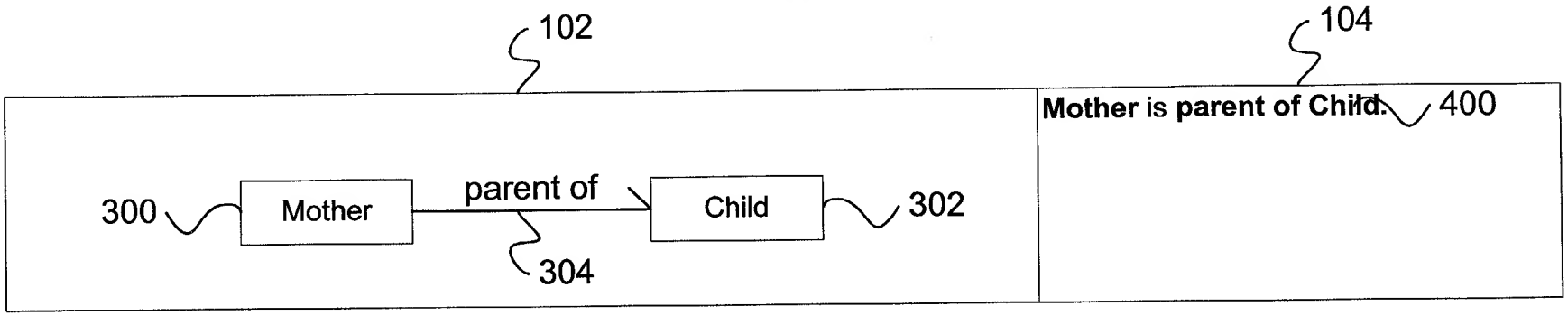


FIG. 14

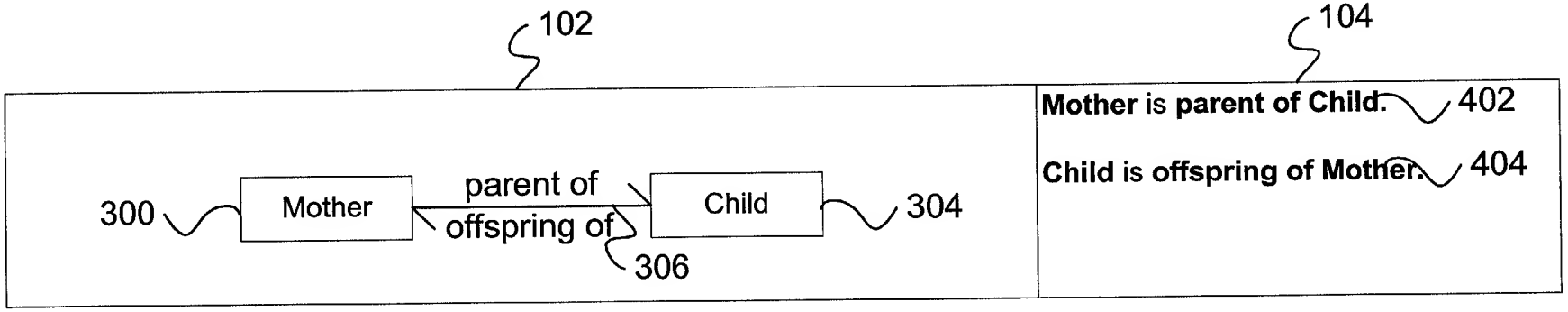


FIG. 15

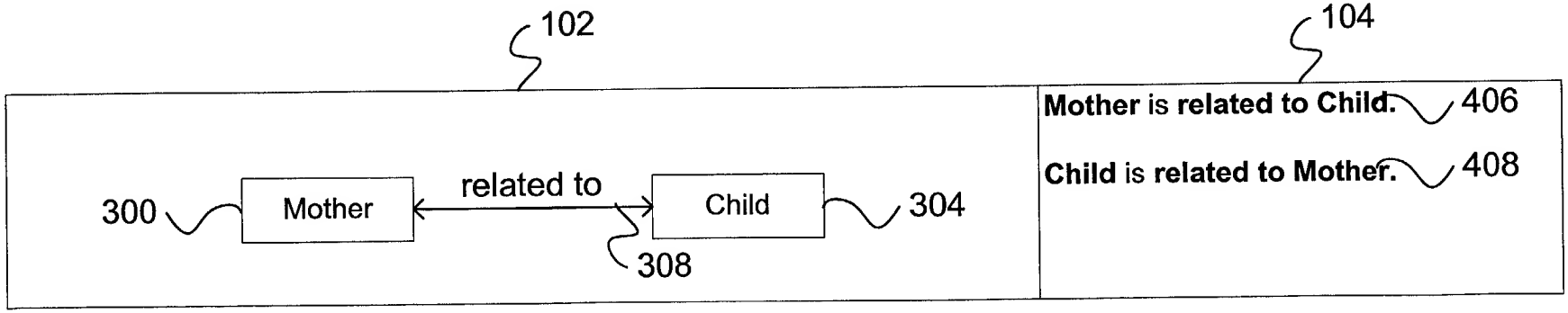


FIG. 16

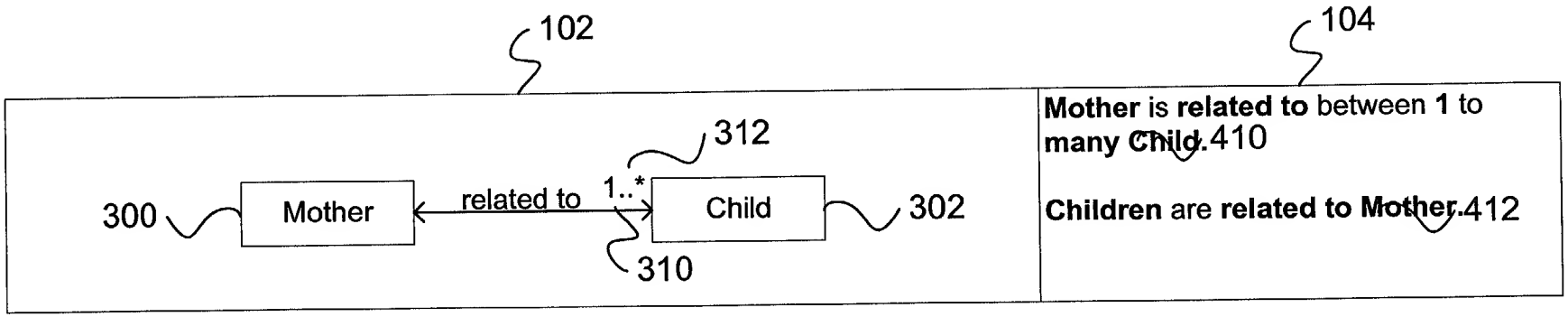


FIG. 17

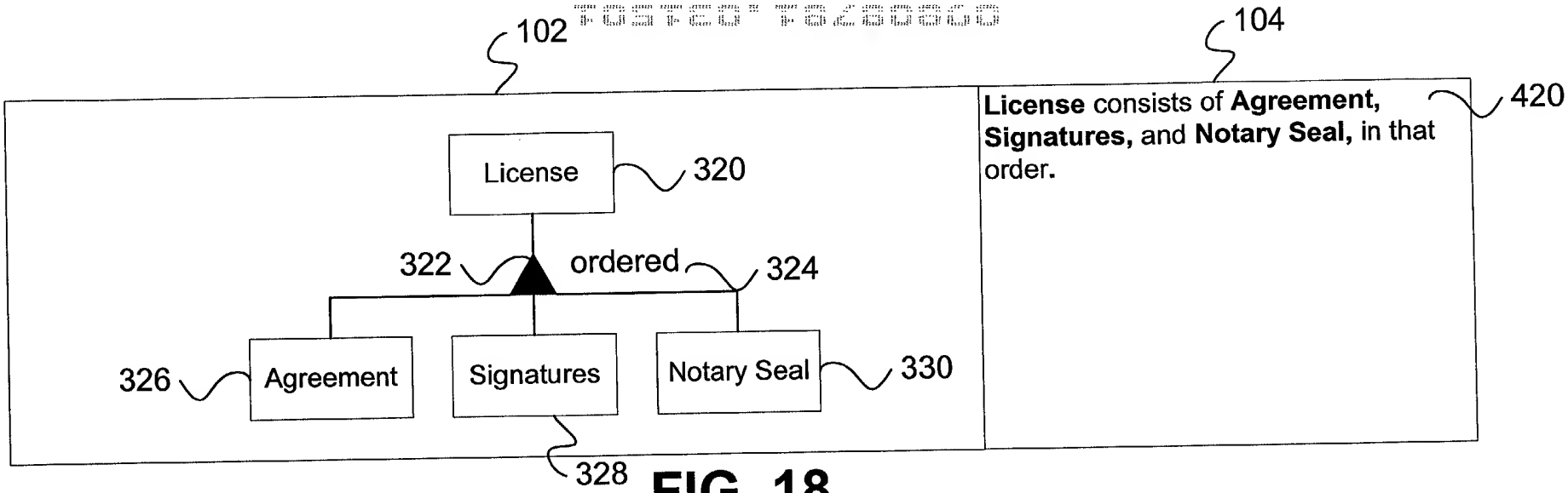

102
104
License
320
322
ordered
324
326
Agreement
Signatures
Notary Seal
330
328
License consists of Agreement, Signatures, and Notary Seal, in that order.
420

FIG. 18

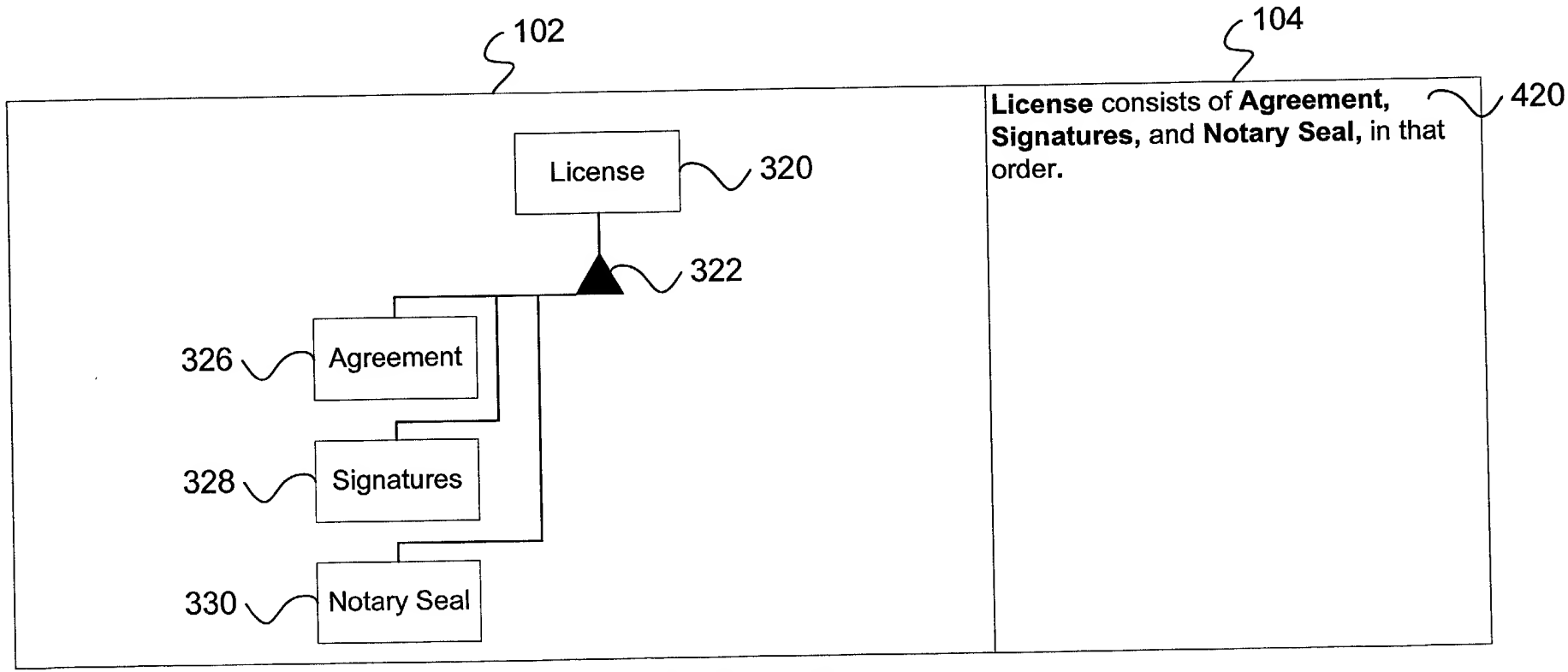

102
104
License
320
322
326
Agreement
328
Signatures
330
Notary Seal
License consists of Agreement, Signatures, and Notary Seal, in that order.
420

FIG. 19

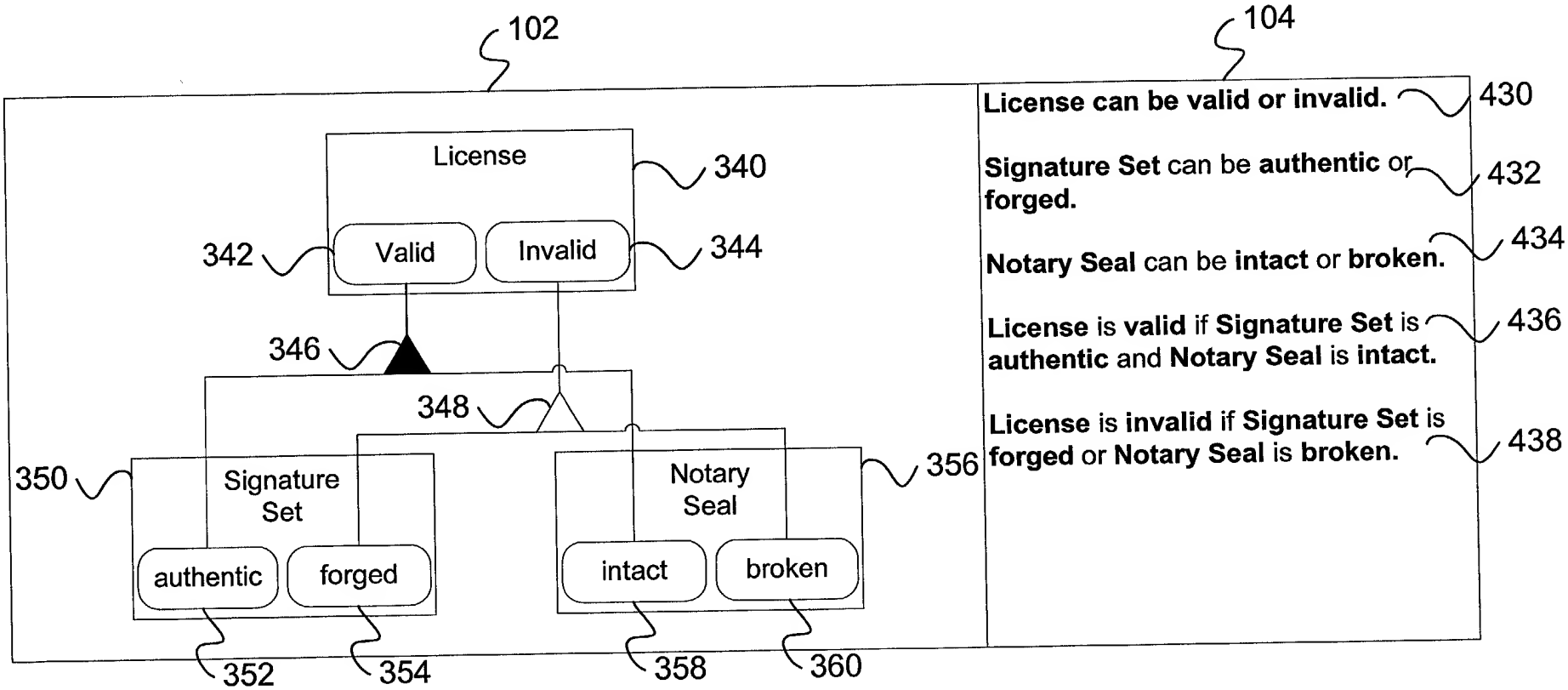


FIG. 20

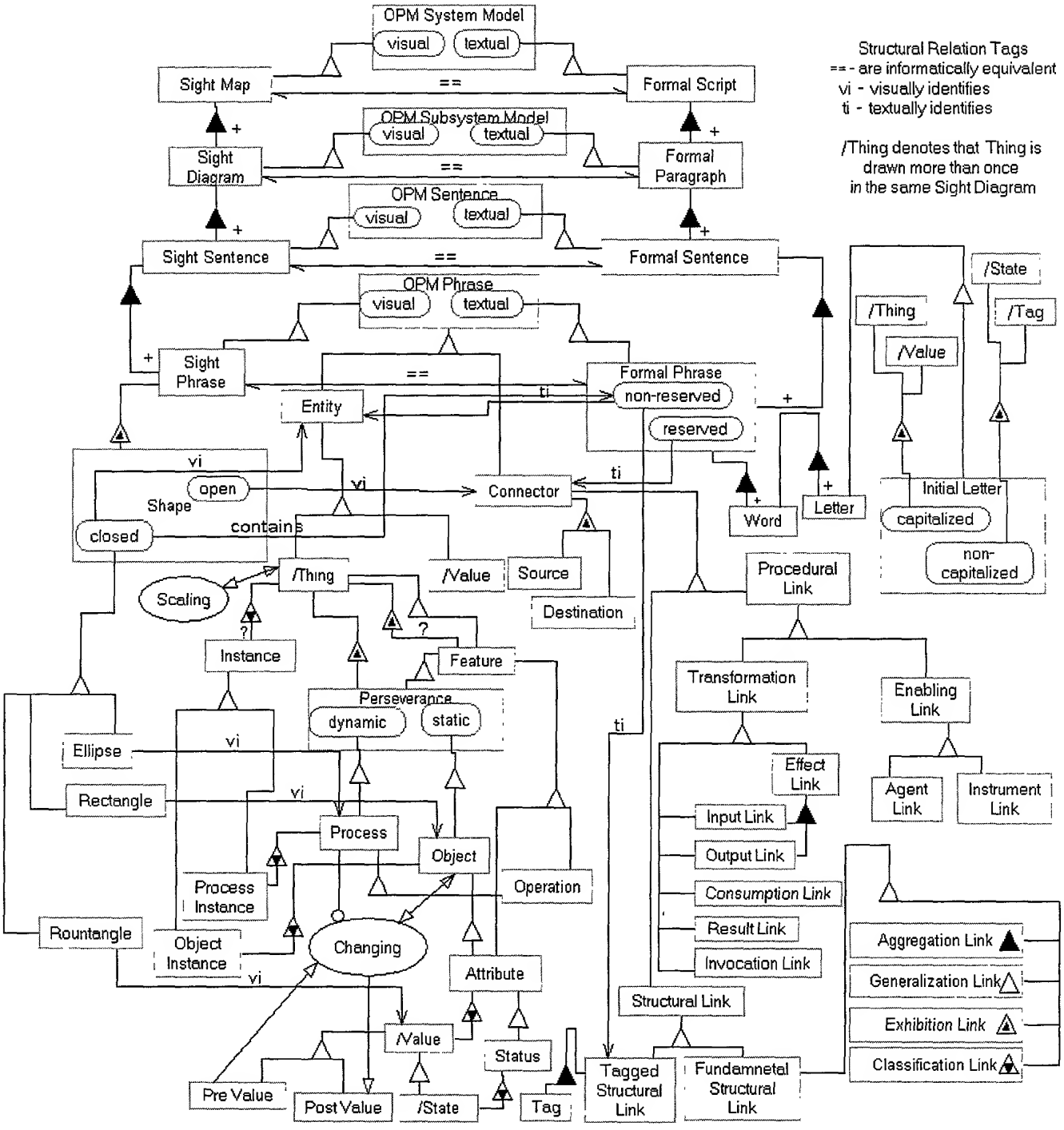

OPM System Model
visual
textual
Sight Map
==
Formal Script
OPM Subsystem Model
visual
textual
Sight Diagram
==
Formal Paragraph
OPM Sentence
visual
textual
Sight Sentence
==
Formal Sentence
OPM Phrase
visual
textual
Sight Phrase
==
Formal Phrase
non-reserved
reserved
Entity
ti
vi
open
Shape
closed
contains
Connector
Scaling
/Thing
/Value
Source
Destination
Instance
Feature
Perseverance
dynamic
static
Ellipse
Rectangle
Process
Object
Process Instance
Operation
Rountangle
Object Instance
Changing
Attribute
/Value
Status
Pre Value
Post Value
/State
Tag
Tagged Structural Link
Fundamnetal Structural Link
Structural Link
Procedural Link
Transformation Link
Enabling Link
Effect Link
Agent Link
Instrument Link
Input Link
Output Link
Consumption Link
Result Link
Invocation Link
Aggregation Link
Generalization Link
Exhibition Link
Classification Link
Word
Letter
/State
/Thing
/Tag
/Value
Initial Letter
capitalized
non-capitalized
Structural Relation Tags
== - are informatically equivalent
vi - visually identifies
ti - textually identifies
/Thing denotes that Thing is drawn more than once in the same Sight Diagram

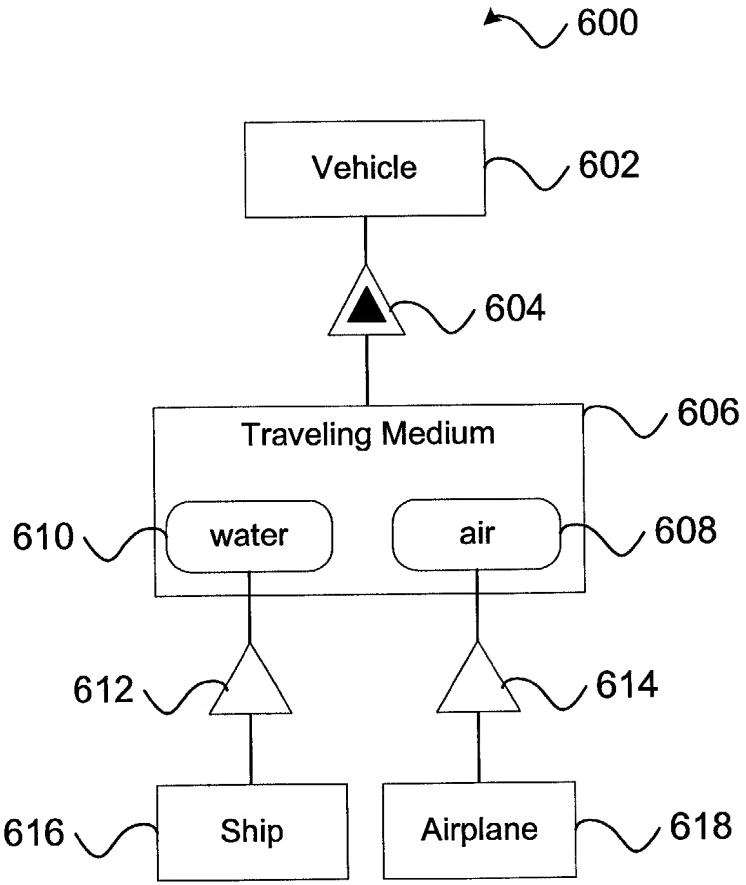
FIG. 21

OPM System Model can be **visual** or **textual.**
Sight Map is a **visual OPM System Model.**
Formal Script is a **textual OPM System Model.**
Sight Map and **Formal Script** are **informatically equivalent.**
Sight Map consists of one or more **Sight Diagrams.**
Formal Script consists of one or more **Formal Paragraphs.**
OPM Subsystem Model can be **visual** or **textual.**
Sight Diagram is a visual OPM Subsystem Model.
Formal Paragraph is a **textual OPM Subsystem Model.**
Sight Diagram and **Formal Paragraph are informatically equivalent.**
Sight Diagram consists of one or more **Sight Sentences.**
Formal Paragraph consists of one or more **Formal Sentences.**
OPM Sentence can be **visual** or **textual.**
Sight Sentence is a **visual OPM Sentence.**
Formal Sentence is a **textual OPM Sentence.**
Sight Sentence and **Formal Sentence** are **informatically equivalent.**
Formal Sentence consists of one or more **Formal Phrases.**
Sight Sentence consists of one or more **Sight Phrases.**
OPM Phrase can be visual or textual.
Sight Phrase is a visual OPM Phrase.
Formal Phrase is a textual OPM Phrase.
Sight Phrase and **Formal Phrase are informatically equivalent.**
Formal Phrase consists of one or more Words.
Word consists of one or more Letters.
Initial Letter is a Letter.
Initial Letter can be **capitalized** or **non-capitalized.**
Thing and **Value** exhibit **capitalized Letter.**
State and **Tag** exhibit **non-capitalized Letter.**
Formal Phrase can be **non-reserved** or **reserved.**
Sight Phrase exhibits **Shape.**
Shape can be **open** or **closed.**
Entity and **Connector** are **OPM Phrases.**
Closed Shape visually identifies Entity.
Non-reserved Formal Phrase textually identifies Entity.
Open Shape visually identifies Connector.

FIG. 22

Reserved Formal Phrase textually identifies **Connector.**
Closed Shape contains **non-reserved Formal Phrase.**
Thing and Value are **Entities.**
Scaling affects **Thing.**
Structural Link and **Procedural Link** are **Connectors.**
Transformation Link and **Enabling Link** are **Procedural Links.**
Connector exhibits **Source** and **Destination.**
Agent Link and **Instrument Link** are **Enabling Links.**
Effect Link, Input Link, Output Link, Consumption Link, Result Link, and Invocation Link are **Transformation Links.**
Effect Link consists of **Input Link** and **Output Link.**
Fundamental Structural Link and **Tagged Structural Link** are **Structural Links.**
Non-reserved Formal Phrase textually identifies **Tagged Structural Link.**
Aggregation Link, Generalization Link, Exhibition Link, and Classification Link are **Fundamental Structural Links.**
Thing instantiates **optional Instances.**
Thing exhibits **optional Features.**
Feature is a **Thing.**
Thing exhibits **Perseverance.**
Perseverance is a **Feature.**
Perseverance can be **static** or **dynamic.**
Object is a **Thing,** the **Perseverance** of which is **static.**
Process is a **Thing,** the **Perseverance** of which is **dynamic.**
Ellipse, Rectangle, and **Rountangle** are **closed Shapes.**
Rectangle visually identifies Object.
Ellipse visually identifies Process.
Attribute and **Operation** are **Features.**
Attribute is an **Object.**
Operation is a **Process.**
Attribute instantiates **Value.**
Rountangle visually identifies Value.
Status is an **Attribute.**
State instantiates **Status.**
State is a **Value.**
Changing affects **Object.**
Changing requires **Process.**
Changing consumes **Pre Value.**
Changing yields **Post Value.**

FIG. 23



Vehicle exhibits **Traveling Medium**. 622

Traveling Medium can be **water** or **air**. 624

Ship is a **Vehicle**, the **Traveling Medium** of which is **water**. 626

Airplane is a **Vehicle**, the **Traveling Medium** of which is **air**. 628

FIG. 24

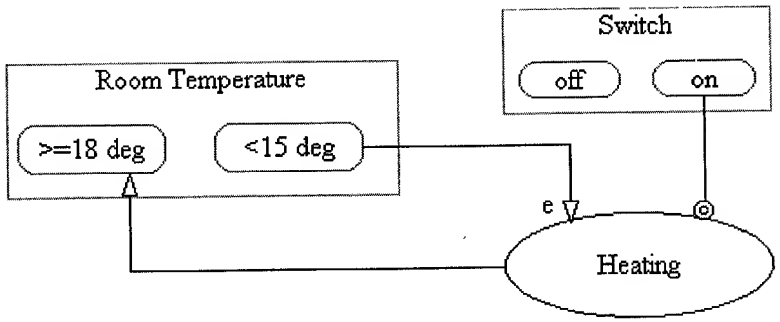


FIG. 25

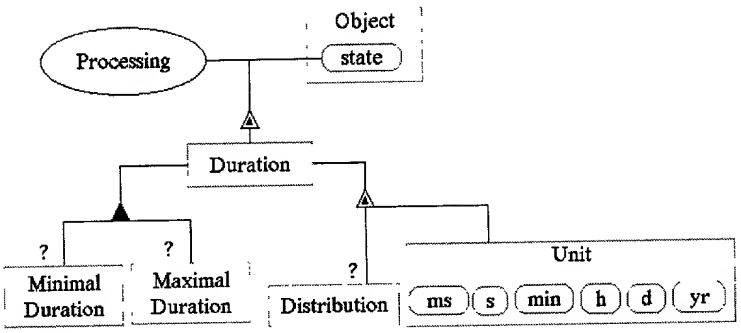

Object
Processing
state
Duration
?
?
?
Minimal Duration
Maximal Duration
Distribution
Unit
ms
s
min
h
d
yr

FIG 26

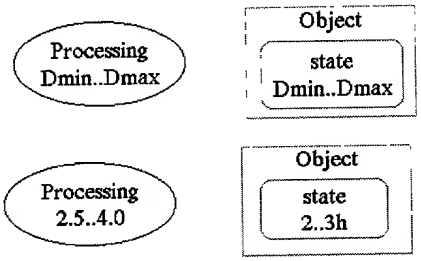
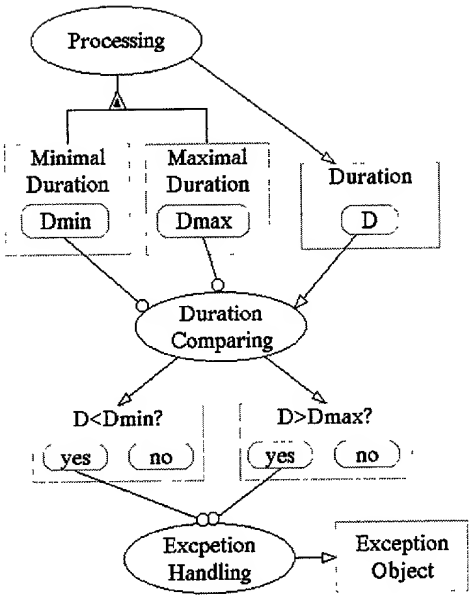

Processing
Dmin..Dmax
Object
state
Dmin..Dmax
Processing
2.5..4.0
Object
state
2..3h

FIG 27



Processing exhibits **Minimal Duration** with value **Dmin**, and **Minimal Duration**, with value **Dmax.**

Processing yields **Duration** with value **D.**

Duration Comparing requires **Dmin** and **Dmax.**

Duration Comparing consumes **D.**

Duration Comparing consumes **D.**

Duration Comparing determines whether **D** is less than **Dmin.**

Duration Comparing determines whether **D** is more than **Dmax.**

Exception Handling occurs if either **D** is more than **Dmax** or **D** is less than **Dmin.**

Exception Handling yields **Exception Object.**

FIG 28

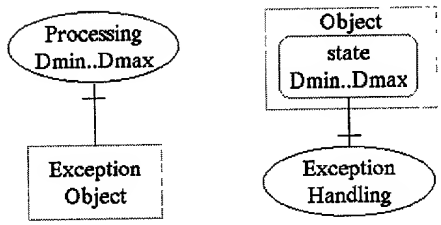

Processing
Dmin..Dmax
Exception
Object
(a)
Object
state
Dmin..Dmax
Exception
Handling
(b)

FIG 29

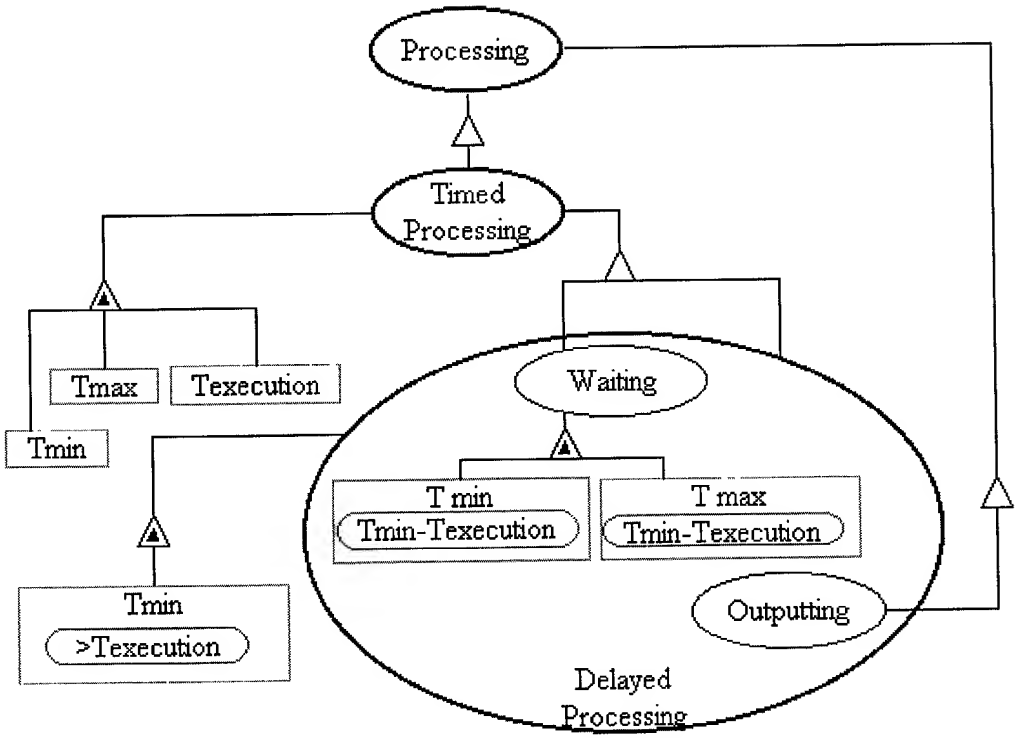
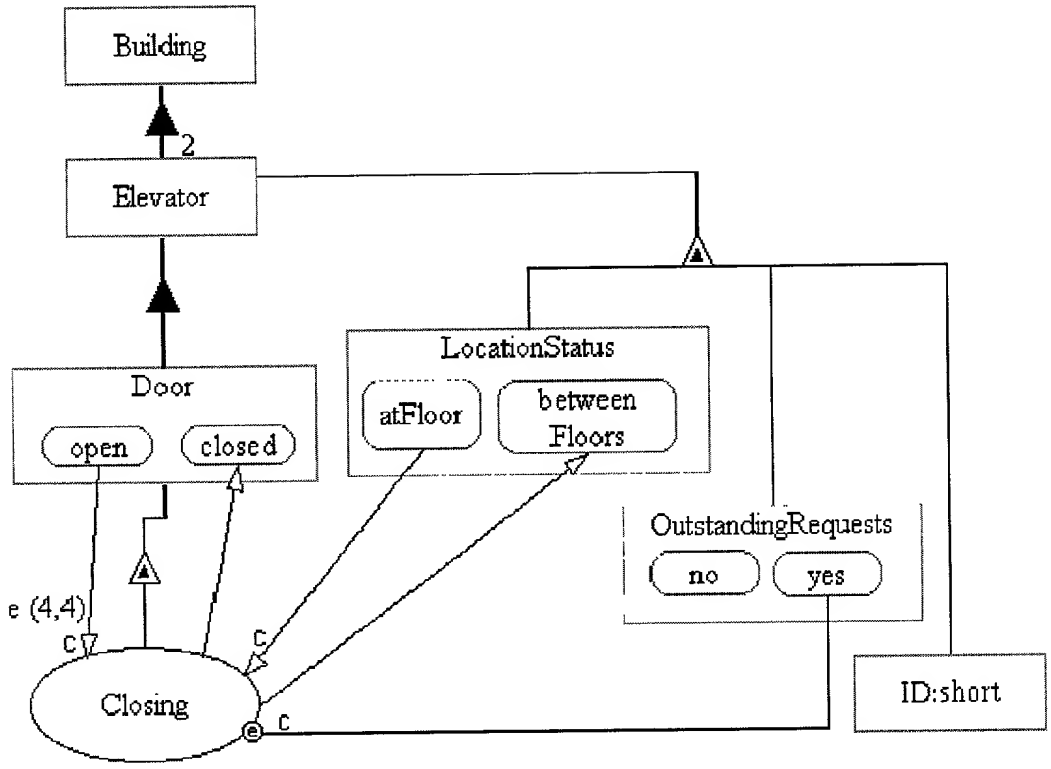


FIG. 30



Object **Building** consists of **2 Elevators**.
Object **Elevator** features **ID**, **OutstandingRequests** and **LocationStatus**.
Object[simple] **ID**:short [key].
Object **OutstandingRequests** has states **no** and **yes**.
Object **LocationStatus** has states **atFloor** and **betweenFloors**.
Elevator consists of **Door**.
Object **Door** has states **open** and **closed**.
Door features process **Closing**.

Event **Door_open** [state-entrance] of **Door** of **Elevator** [**ID** in {**1** to **2**}] of **Building** triggers **Closing** of **Door** of **Elevator**[**ID**] of **Building** with a reaction time of (4,4).

Process **Closing** is guarded by "**Door** is **open**",
"**OutstandingRequests** of **Elevator** is **yes**" and
"**LocationStatus** of **Elevator** is **atFloor**".
Closing affects **Door** of **Elevator** from **open** to **closed** and
LocationStatus of **Elevator** from **atFloor** to **betweenFloors**.

FIG. 31

```
1 class Building : public Object {
2 private:
3   class Elevator : public Object {
4   private:
5     Building *f;
6     short ID;
7   public:
8     short get_ID(void);
9     void set_ID(short t);
10  private:
11    class Door : public Object {
12    private:
13      Elevator *f;
14      State* closed;     // same for State open:  ...
15    public:
16      void set_closed(BOOLEAN enter, short Invoker);
17      void enter_closed(short Invoker);
18      void enter_closed(void);
19      enum door get_Door(void);
20      friend Elevator;
21    private:
22      class Closing : public Process {
23      private:
24        Door *f;
25        void Execute(short Invoker, short Triggering_Event);
26      public:
27        Closing(void);
28        BOOLEAN Trigger(short Invoker, short
29              Triggering_event);
30        friend Door;
31      };
32   public:
33      Closing the_Closing;
34      friend Closing;
35      Door(void);
36   };
37  public:
38    Door the_Door;
// LocationStatus and OutstandingRequests are defined in
// a way similar to Door
...
39    Elevator(void);
40 };
41 public:
42   Elevator the_Elevator[2];
43   Building(void);
44 };
45 extern Building the_Building;
```

Elevator features simple object ID

Elevator consists of Door

Door features Process Closing

FIG. 32

```
BOOLEAN Building::Elevator::Door::Closing::Trigger(short Invoker, short
Triggering_event)
{
  BOOLEAN result = TRUE;

  if (f->get_Door()==OPEN)
      result = TRUE; // if the Elevator Door is OPEN then
                     // proceed to check if there are
                     // OutstandingRequests for the Elevator
  else
      result = FALSE;
  if (result == TRUE)
  {
      if
       (f->f->the_OutstandingRequests.get_OutstandingRequests()==YES)
          result = TRUE;  // if there are OutstandingRequests then
                          // proceed to check if Elevator is ATFLOOR
      else
          result = FALSE;
      }
      if (result == TRUE)
      {
         if (f->f->the_LocationStatus.get_LocationStatus()==ATFLOOR)
            result = TRUE; // if if Elevator is ATFLOOR then
                           // proceed to open the Elevator Door
         else
            result = FALSE;
      }
      if (result == TRUE)
      {
         Execute(Invoker, Triggering_event); // open the Door
         return TRUE;
      }
      else
         return FALSE;
}
```

FIG. 33

```
void _EventProcessTable::TriggerAllProcesses(Event *e)
{
   Process *process_ptr;

   switch(e->get_id())      // get id of triggering_Event
     // (e.g., door_open)
   {
      case dooropen:  switch (e->get_trig_obj()){
                      // get id of triggering object
         case elevator1:
           process_ptr =
            &the_Building.the_Elevator[0].the_Door.the_Closing;
         break;
         case elevator2:
           process_ptr =
            &the_Building.the_Elevator[1].the_Door.the_Closing;
         break;
      }
      TryToTrigger(e,process_ptr, 0,4,0,0,0,4,0,0,
           e->get_trig_obj(), e->get_id());
      break;
   }
}
```

FIG. 34

```
void Building::Elevator::Door::Closing::Execute(short Invoker,
short Triggering_event)
{
      Process::Execute(Invoker, Triggering_Event);
      f->f->the_Door.enter_closed(Invoker);
      f->f->the_LocationStatus.enter_betweenFloors();
}
```

FIG. 35

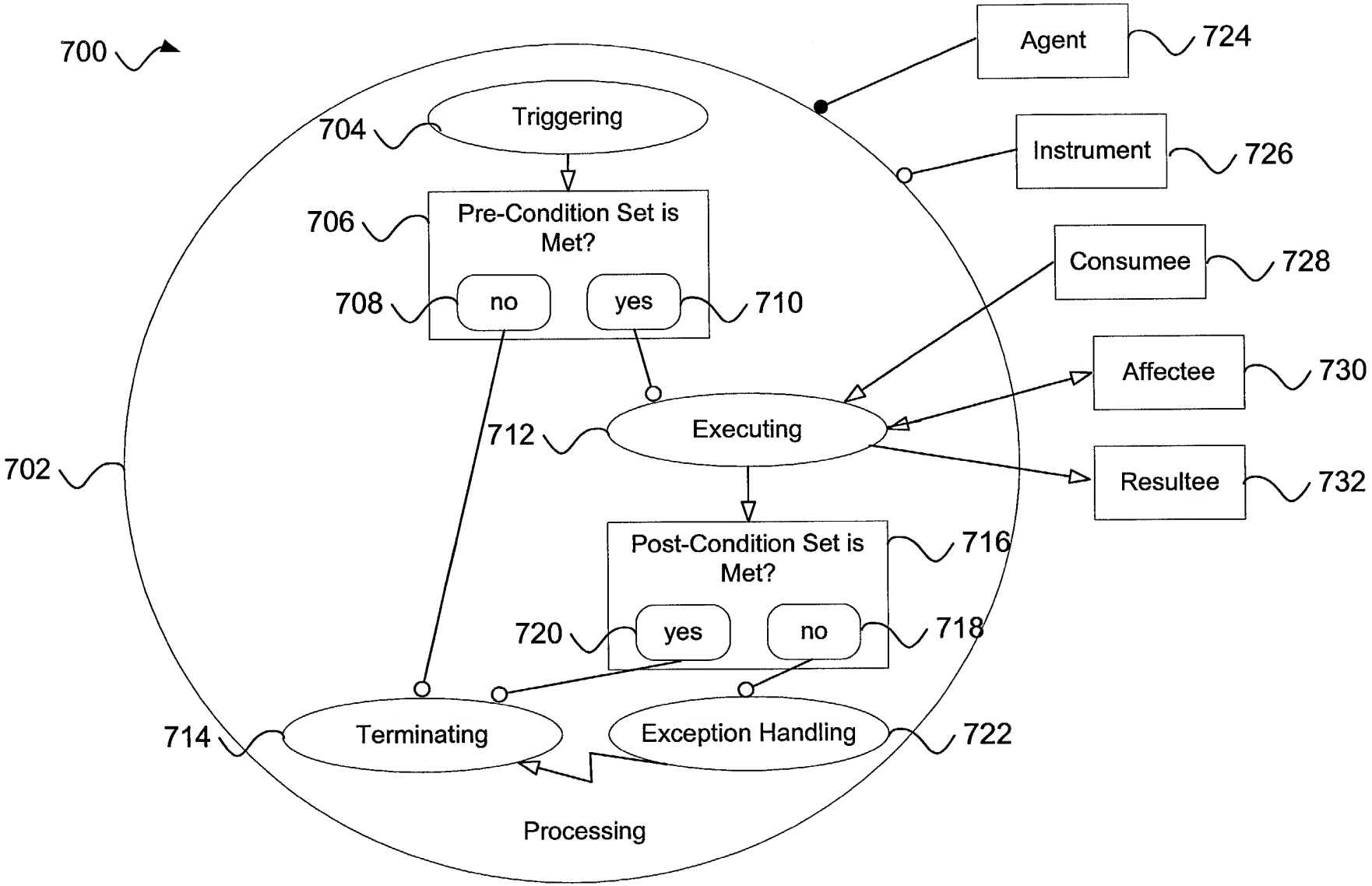

700
Agent
724
Triggering
704
Instrument
726
Pre-Condition Set is Met?
706
Consumee
728
no
yes
708
710
Affectee
730
712
Executing
702
Resultee
732
Post-Condition Set is Met?
716
720
yes
no
718
714
Terminating
Exception Handling
722
Processing

FIG. 36

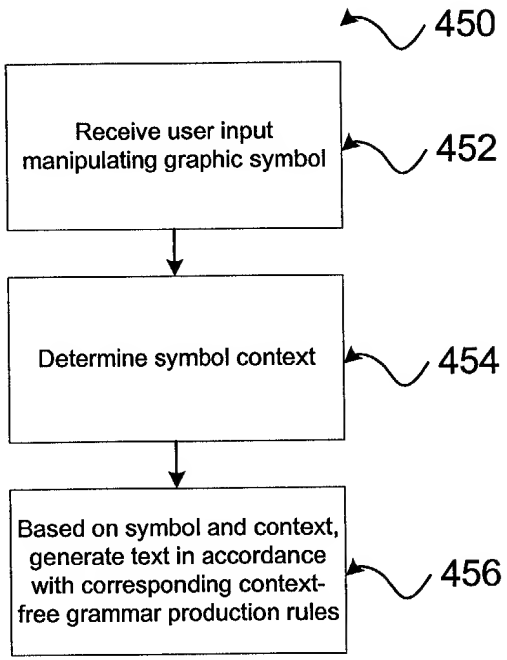

450
Receive user input manipulating graphic symbol
452
Determine symbol context
454
Based on symbol and context, generate text in accordance with corresponding context-free grammar production rules
456

FIG. 37

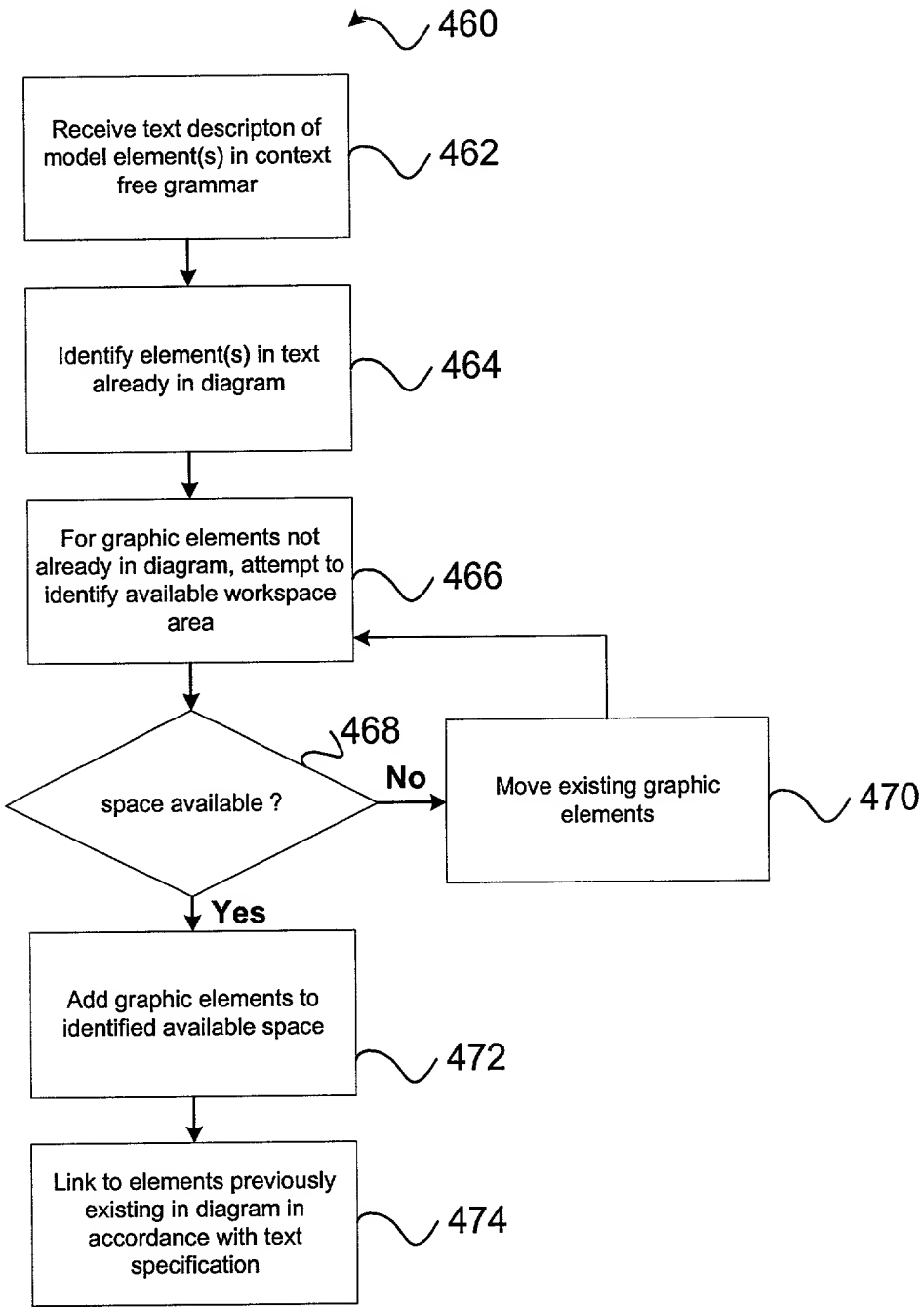

460
Receive text descripton of model element(s) in context free grammar
462
Identify element(s) in text already in diagram
464
For graphic elements not already in diagram, attempt to identify available workspace area
466
468
space available ?
No
Move existing graphic elements
470
Yes
Add graphic elements to identified available space
472
Link to elements previously existing in diagram in accordance with text specification
474

FIG. 38

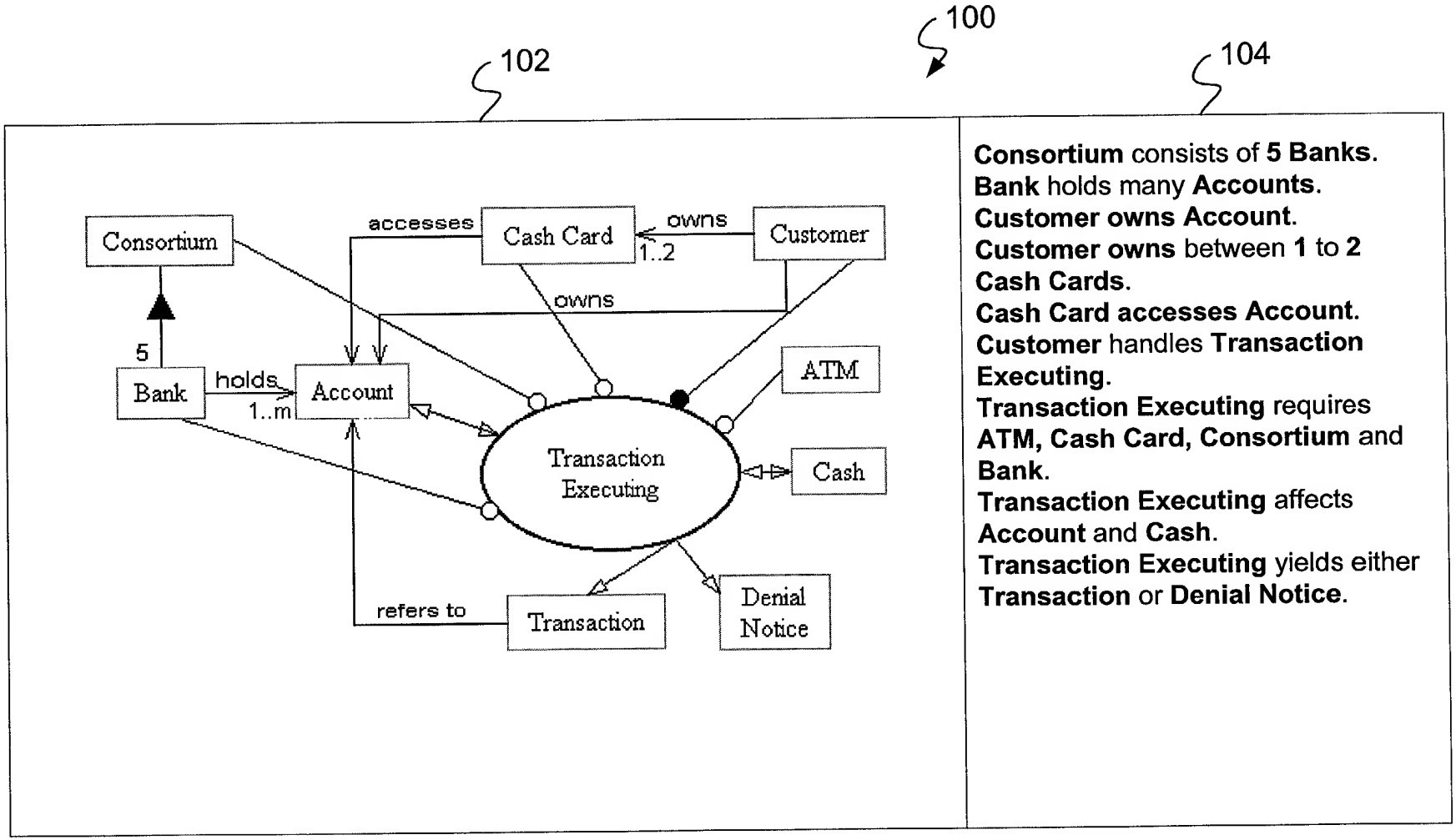

100
102
104
Consortium
accesses
Cash Card
owns
1..2
Customer
owns
5
Bank
holds
1..m
Account
ATM
Transaction
Executing
Cash
refers to
Transaction
Denial
Notice
Consortium consists of 5 Banks.
Bank holds many Accounts.
Customer owns Account.
Customer owns between 1 to 2 Cash Cards.
Cash Card accesses Account.
Customer handles Transaction Executing.
Transaction Executing requires ATM, Cash Card, Consortium and Bank.
Transaction Executing affects Account and Cash.
Transaction Executing yields either Transaction or Denial Notice.

FIG. 39

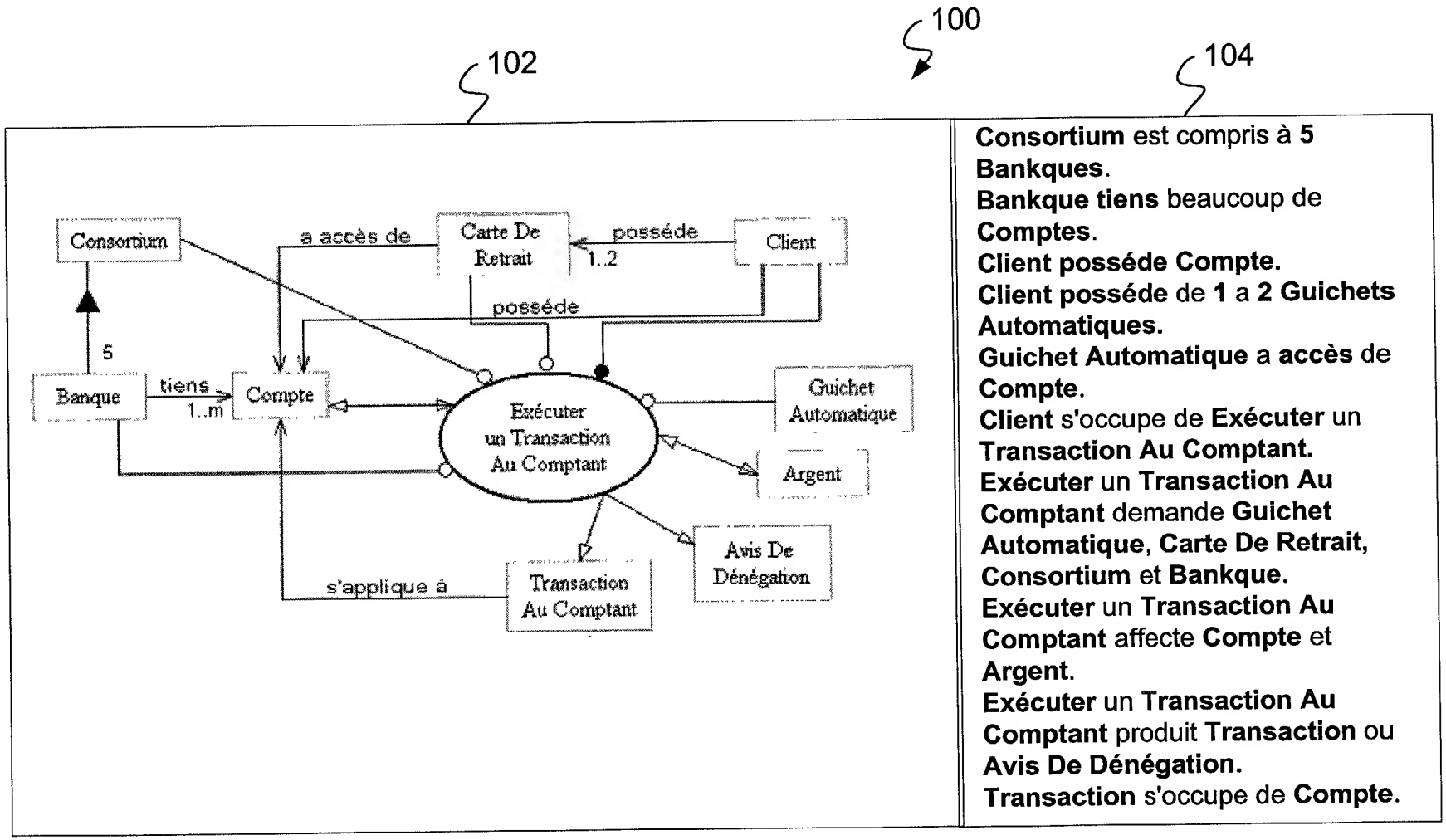


FIG. 40

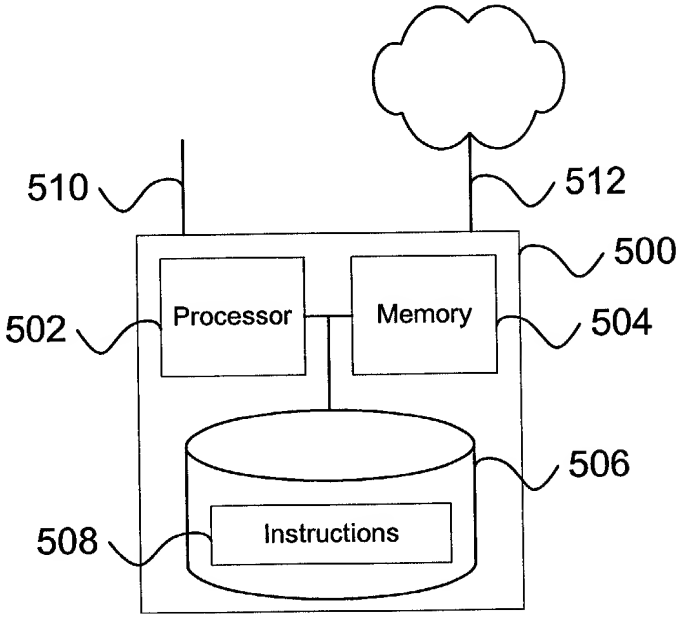


FIG. 41